
ns-3 Training

ns-3 Annual Meeting
June 2017

ns-3 training goals

- Make attendees more productive with ns-3
 - Learn about the project scope, and where to get additional help
 - Understand the architecture and design goals of the software
 - Introduce how to write new code for the simulator
 - Learn about selected topics in more detail
 - Answer your questions

Agenda and Instructors

- Software and usage overview (T. Henderson)
- How to write new models (T. Pecorella)
- Wi-Fi and wireless models (T. Henderson)
- TCP and AQM models (M. Tahiliani)
- Traffic control (S. Avallone)

Please ask questions along the way!

Additional training archives

- LTE (Lorenza Giupponi and Biljana Bojovic), June 2016
- Parallel, Distributed Simulations (Peter Barnes), June 2016
- Direct Code Execution (Tom Henderson), June 2016
- Tracing (Walid Younes), June 2014

<http://www.nsnam.org/wiki/Training2017>

Your feedback on requested topics

1) what is your past level of experience with ns-3?

- various (from starting the tutorial to having written new models)

2) what technical topics in the simulator interest you the most?

- Wi-Fi, LTE, TCP
- routing 6LoWPAN, IoT, IPv6, BGP, and the core

3) past level of experience with any other network simulation tools?

- MATLAB/Simulink, plus ns-2, OPNET, OMNeT++, Totem

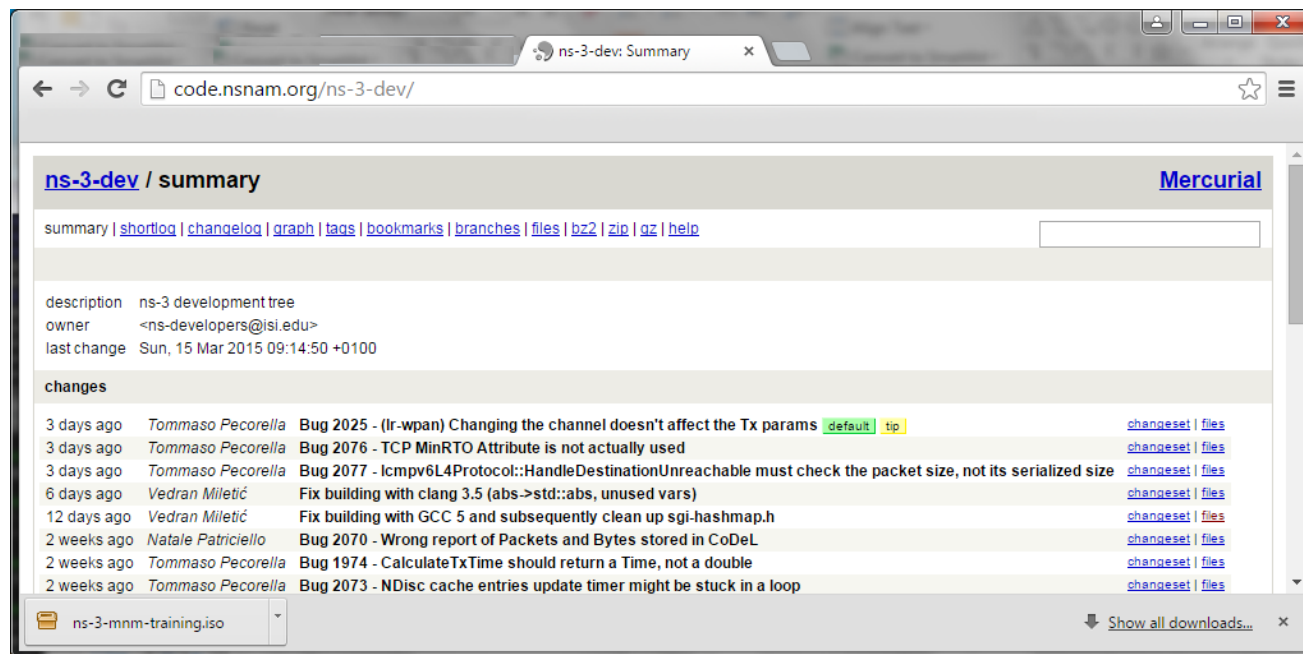
Your feedback on requested topics

4) what do you most want to get out of the training sessions

- refresh, get ideas for lab assignments, understand real-time simulations, inject real traffic, global tips and tricks about ns-3, learn LTE, implement new models

Options for working along

- 1) Download the required packages onto your (Linux, OS X, or BSD) system
- 2) Download the ISO image (Live DVD)
- 3) Browse the code online: <https://code.nsnam.org>



Project overview

Motivations for ns-3 project

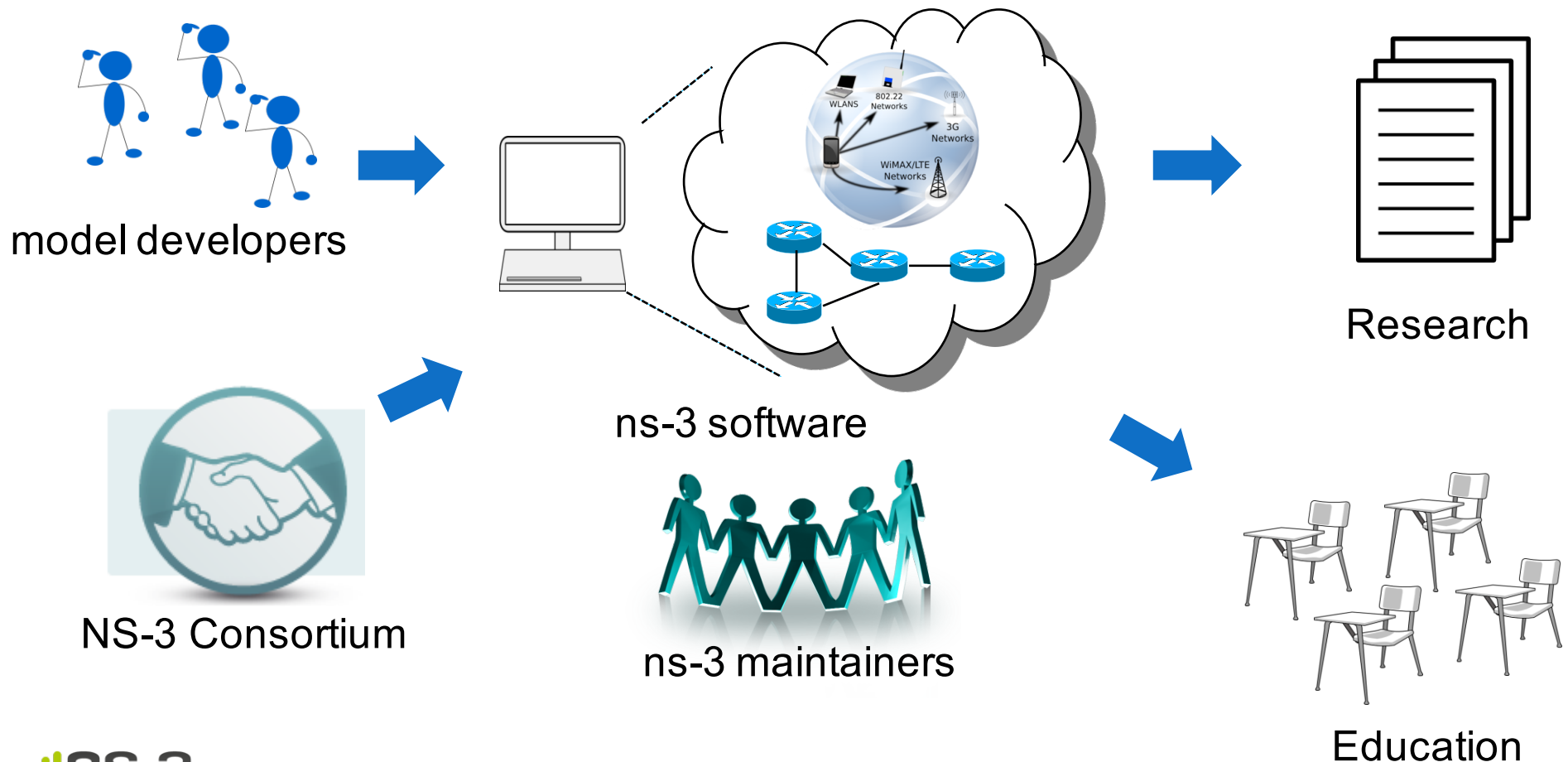
Develop an extensible simulation environment for networking research

- 1) a tool **aligned with the experimentation needs** of modern networking research
- 2) a tool that **elevates the technical rigor** of network simulation practice
- 3) an **open-source project** that encourages community contribution, peer review, and long-term maintenance and validation of the software

Community-maintained, scientific computing software by following best current practices for open source

ns-3: An Open Source Network Simulator

- ns-3 is a **discrete-event network simulator** targeted for **research and educational use**



What have people done with ns-3?

- thousands of publications to date
- search of 'ns-3 simulator' on IEEE and ACM digital libraries, or Google Scholar

FSR: Formal Analysis and Implementation Toolkit for Safe Interdomain Routing

Anduo Wang, Lixian Jia, Member, IEEE, Wenchao Zhou, Yiqing Ren, Boon Thau Loo, Jennifer Redford, Senior Member, IEEE, Vivek Nigam, Andre Sedrov, and Carolyn Talbot

Abstract—Interdomain routing stitches the disparate parts of the Internet together, making protocol stability a critical issue to both researchers and practitioners. Yet, researchers create safety proofs and counterexamples by hand and build simulators and prototypes to explore protocol dynamics. Similarly, network operators analyze their router configurations manually or using languages tools. In this paper, we present a comprehensive toolkit for analyzing and implementing routing policies, ranging from high-level guidelines to specific router configurations. Our *Formally Safe Routing (FSR)* toolkit performs all of these functions from the same algebraic representation of routing policy. We show that routing algebra has a natural translation to both integer constraints (to perform safety analysis with SAT solvers) and declarative programs (to generate distributed implementations). Our extensive experiment with realistic topologies and policies shows how FSR can detect problems in an autonomous system's (AS's) BGP configuration, prove sufficient conditions for Border Gateway Protocol (BGP) safety, and empirically evaluate convergence time.

Index Terms—Communications technology, declarative networking, formal analysis, routing algebra.

I. INTRODUCTION

THE INTERNET'S global routing system does not necessarily converge, depending on how the Border Gateway Protocol (BGP) policies of individual networks are configured. Since protocol oscillations cause serious performance disruptions and router overload, researchers devote significant attention to BGP stability (or "safety"). Abstract formal models of BGP [12]–[15] allow researchers to explore how local policies affect BGP stability and identify policy guidelines that, if universally adopted by ISPs, ensure global

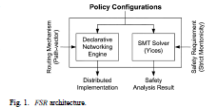


Fig. 1. FSR architecture

convergence [4], [8]–[11], [33]. While our understanding of BGP safety has improved dramatically in the past decade, each research study still proceeds independently—manually creating proofs and counterexamples, and sometimes building simulators or prototypes to study protocol overhead and transient behavior during convergence.

To aid the design, analysis, and evaluation of safe interdomain routing, we propose the *Formally Safe Routing (FSR)* toolkit. FSR serves two important communities. For researchers, FSR automates important parts of the design process and provides a common framework for describing, evaluating, and comparing new safety guidelines. For network operators, FSR automates the analysis of internal router (iBGP) and border gateway (eBGP) configurations for safety violations. For both communities, FSR automatically generates realistic protocol implementations to evaluate real network configurations (e.g., to study convergence time) prior to actual deployment. The ideas underlying FSR also study research in routing algebra [13], [16] with recent advances in declarative networking [22] to produce provably correct implementations of safe interdomain routing.

Given policy configurations as input, FSR produces an analysis of safety properties and a distributed protocol implementation, as shown in Fig. 1. FSR has three main underlying technologies.

- **Policy configuration as algebra:** Our extension to routing algebra [12], [36] allows researchers and network operators to express policy configurations in an abstract algebraic form. These configurations can be anything from high-level policy guidelines (e.g., proposed constraints that a researcher wants to study) or a completely specified policy instance (e.g., an iBGP configuration of a multi- autonomous-system (AS) network that an operator wants to analyze). Router configuration files can be automatically translated into the algebraic representation, easing the adoption of FSR.
- **Safety analysis:** To automatically analyze the policy configuration, FSR reduces the convergence proof to a

Wireless Netw. (2013) 17:1775–1794
DOI 10.1007/s11276-011-0077-0

Message delivery in heterogeneous networks prone to episodic connectivity

Rao Naveed Bin Rais · Thierry Turetli · Kati Orazcka

Published online: 17 August 2011
© Springer Science+Business Media, LLC 2011

Abstract We present an efficient message delivery framework, called *MedHea*, which enables communication in an internet connecting heterogeneous networks that is prone to disruptions in connectivity. *MedHea* is complementary to the IRTTP's Bundle Architecture; besides its ability to store messages for unavailable destinations, *MedHea* can bridge the connectivity gap between infrastructure-based and multi-hop infrastructure-less networks. It benefits from network heterogeneity (e.g., nodes supporting more than one network and nodes having diverse resources) to improve message delivery. For example, in IEEE 802.11 networks, participating nodes may use both infrastructure- and ad-hoc modes to deliver data to otherwise unavailable destinations. It also employs opportunistic routing to support nodes with episodic connectivity. One of *MedHea*'s key features is that any *MedHea* node can relay data to any destination and can act as a gateway to make two networks inter-operate or to connect to the backbone network. The network is able to store data destined to temporarily unavailable nodes till the time of their expiry. This time period depends upon current storage availability as well as quality-of-service needs (e.g., delivery delay bounds) imposed by the application. We showcase

MedHea's ability to operate in environments consisting of a diverse set of interconnected networks and evaluate its performance through extensive simulations using a variety of scenarios with realistic synthetic and real mobility traces. Our results show significant improvement in average delivery ratio and a significant decrease in average delivery delay in the face of episodic connectivity. We also demonstrate that *MedHea* supports different levels of quality-of-service through traffic differentiation and message prioritization.

Keywords Disruption tolerance · Episodic connectivity · Heterogeneous networks · Node relaying · Store-carry-and-forward · DTN routing

1 Introduction

It is envisioned that the Internet of the future will be highly heterogeneous not only due to the wide variety of end devices it interconnects, but also in terms of the underlying networks it comprises. Figure 1 illustrates networks that range from wired and wireless backbones (e.g., community wireless mesh networks) to wireless infrastructure-based and ad-hoc networks (e.g., MANETs). On the other hand, current and emerging applications, such as emergency response, environmental monitoring, smart environments (e.g., smart offices, homes, museums, etc.), and vehicular networks, among others, increasingly require and arbitrarily long-lived disruptions in connectivity. The resulting disruption- or delay-tolerant networks (DTNs) will likely become an important component of future networks.

Seamless interoperability among heterogeneous networks is a challenging problem as these networks may have very different characteristics. Node diversity may also

Augmenting Data Center Networks with Multi-Gigabit Wireless Links

Daniel Halperin¹, Jitendra Kandula¹, Jitendra Padhye¹, Paramvir Bahl¹, and David Wetherall¹
Microsoft Research¹ and University of Washington²

Abstract—The 60 GHz wireless technology that is now emerging has the potential to provide dense and extremely fast connectivity at low cost. In this paper, we explore its use to relieve bottlenecks in data center (DC) networks. By experimenting with prototype equipment, we show that the DC environment is well suited to a deployment of 60 GHz links contrary to concerns about interference and link reliability. Using directional antennas, many wireless links can run concurrently at multi-Gbps rates on top-of-rack (ToR) switches. The wired DC network can be used to sidestep several common wireless problems. By analyzing production traces of DC traffic for four real applications, we show that adding a small amount of network capacity in the form of *wireless flyways* is the wired DC network can improve performance. However, to be of significant value, we find that one hop indirect routing is needed. Informed by our 60 GHz experiments and DC traffic analysis, we present a design that uses DC traffic levels to select and add flyways to the wired DC network. Trace-driven evaluations show that network limited DC applications with predictable traffic workloads running on a 1:2 oversubscribed network can be sped up by 45% in 95% of the cases, with just one wireless device per ToR switch. With two devices, in 40% of the cases, the performance is identical to that of a non-oversubscribed network.

Traditional, wired DC networks are tree-structured and oversubscribed to keep costs down [13]. For example, a typical DC rack comprises 40 machines connected to a top-of-the-rack (ToR) switch with 1 Gbps links. The ToR is connected to an aggregation switch (to network with other racks) with 10 Gbps links. Thus, the link from the ToR to the aggregation switch can be oversubscribed with a ratio of 1:4. However, each oversubscribed link is a potential bottleneck that hinders some DC application. Recent research tackles this problem by combining many more links and switches with variants of multipath routing so that the core of the network is no longer oversubscribed [1, 8, 9]. Of course, this benefit comes with large material cost and implementation complexity [15]. Some designs require so many wires that cabling becomes a challenge [1], and most require "fork lift" [3] upgrades to the entire infrastructure.

In prior work [15], we argued instead for a more modest addition of links to relieve bottlenecks and boost application performance. The links, called *wireless flyways*, add extra capacity to the base network to alleviate bottlenecks. When the traffic matrix is sparse (i.e., only a few flyways are used), a small number of flyways can significantly improve performance, without the cost of building a fully non-oversubscribed network.

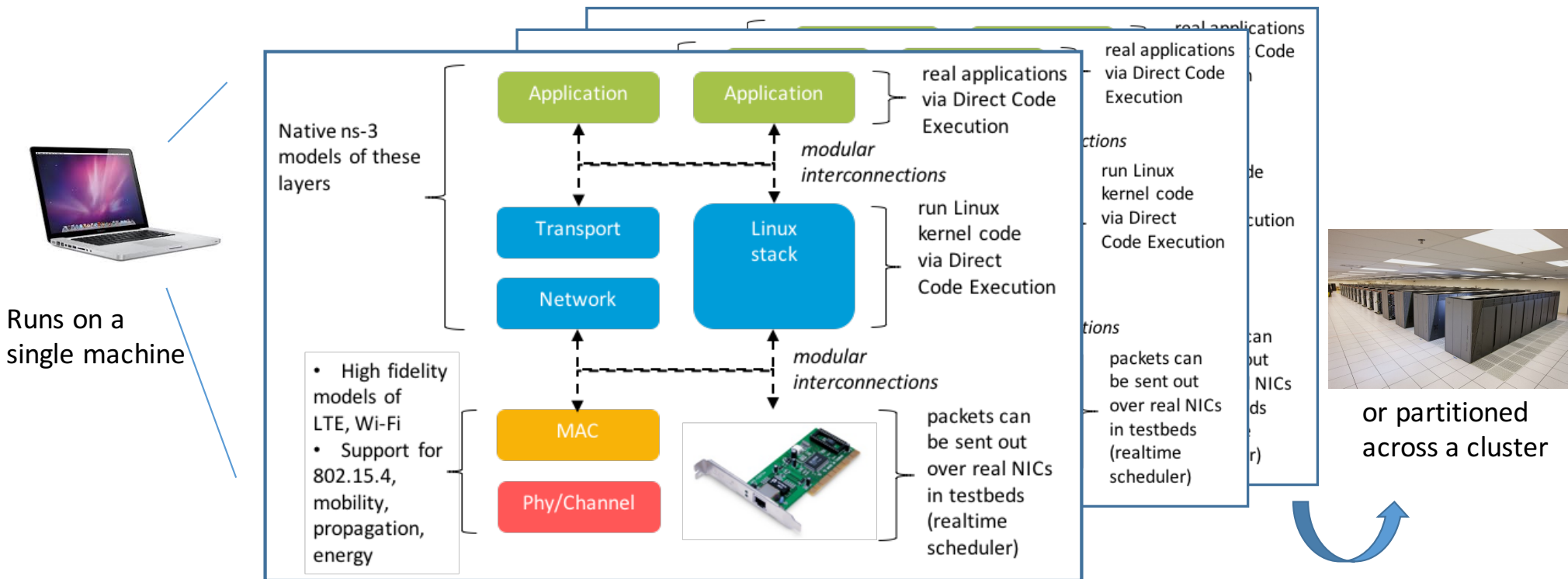
The basic design of a DC network with 60 GHz flyways is as follows. The *wired network* is provisioned for the average case and can be oversubscribed. Each top-of-rack (ToR) switch is equipped with one or more 60 GHz wireless devices, with electronically steerable directional antennas. A central controller monitors DC traffic patterns, and switches the beams of the wireless devices to set up flyways between ToR switches that provide added bandwidth to needed.

Other researchers have explored use of fiber optic cables and MEMS switches [7, 30] for creating flyways. We believe that 60 GHz flyways are an attractive choice because wireless devices simplify DC upgrades, as no wiring changes are needed. Furthermore, 60 GHz technology is likely to become inexpensive as it is commoditized by consumer applications, while optical switches are not. Wireless devices can introduce additional issues as well—for example, with dynamic topologies, the network management may become more difficult.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear the name and the full citation on the first page. To copy otherwise, to republish, to post on server or to redistribute to lists, requires prior specific permission and/or a fee.
© 2011 ACM 978-1-4503-0797-0/11/0000...\$10.00.
Copyright 2011 ACM 978-1-4503-0797-0/11/0000...\$10.00.

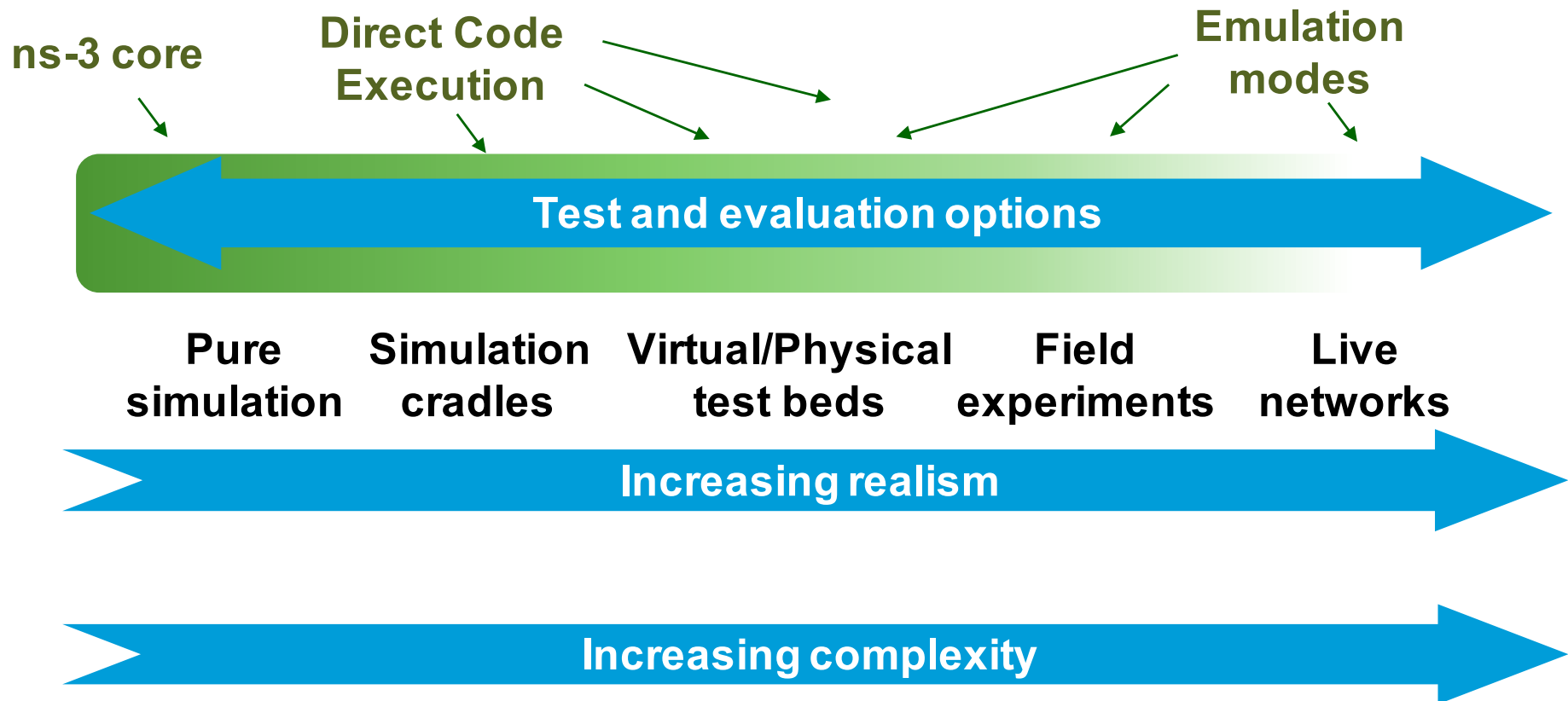
ns-3 overview

- ns-3 is a leading open source, **packet-level network simulator** oriented towards network research, featuring a **high-performance core** enabling **parallelization across a cluster** (for large scenarios), **ability to run real code**, and **interaction with testbeds**



Network performance evaluation options

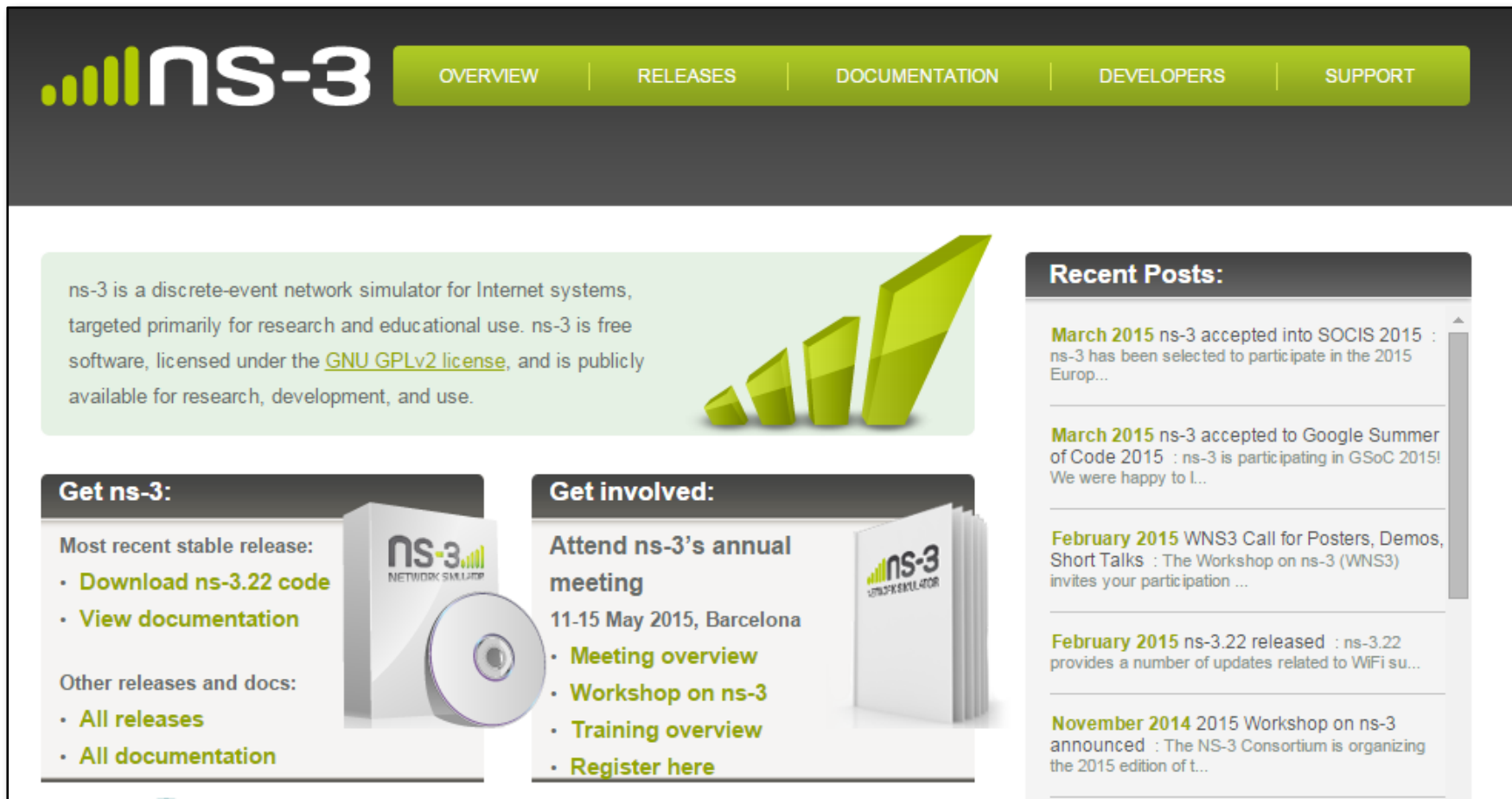
- ns-3 enables researchers to more easily move between simulations, test beds, and experiments



-
- The open-source project

ns-3 main website

- Project home: <https://www.nsnam.org>



The screenshot shows the ns-3 main website. At the top is a dark header with the ns-3 logo on the left and a green navigation bar with links: OVERVIEW, RELEASES, DOCUMENTATION, DEVELOPERS, and SUPPORT. Below the header, the main content area has a light green background. On the left, a text box describes ns-3 as a discrete-event network simulator for Internet systems, targeted for research and educational use, free software under the GNU GPLv2 license, and publicly available for research, development, and use. To the right of this text is a 3D bar chart with four bars of increasing height. Below the text box are two columns. The left column, titled 'Get ns-3:', lists the most recent stable release (ns-3.22) with links to download the code and view documentation, and other releases and docs (all releases, all documentation). The right column, titled 'Get involved:', lists the annual meeting (11-15 May 2015, Barcelona) with links to meeting overview, workshop on ns-3, training overview, and register here. On the far right, a 'Recent Posts:' section lists four posts: March 2015 ns-3 accepted into SOCIS 2015, March 2015 ns-3 accepted to Google Summer of Code 2015, February 2015 WNS3 Call for Posters, Demos, Short Talks, and February 2015 ns-3.22 released. The bottom of the page features the ns-3 logo and the text 'ns-3 Training, June 2017'.

ns-3 is a discrete-event network simulator for Internet systems, targeted primarily for research and educational use. ns-3 is free software, licensed under the [GNU GPLv2 license](#), and is publicly available for research, development, and use.

Get ns-3:

Most recent stable release:

- [Download ns-3.22 code](#)
- [View documentation](#)

Other releases and docs:

- [All releases](#)
- [All documentation](#)

Get involved:

Attend ns-3's annual meeting

11-15 May 2015, Barcelona

- [Meeting overview](#)
- [Workshop on ns-3](#)
- [Training overview](#)
- [Register here](#)

Recent Posts:

March 2015 ns-3 accepted into SOCIS 2015 : ns-3 has been selected to participate in the 2015 Europ...

March 2015 ns-3 accepted to Google Summer of Code 2015 : ns-3 is participating in GSoC 2015! We were happy to L...

February 2015 WNS3 Call for Posters, Demos, Short Talks : The Workshop on ns-3 (WNS3) invites your participation ...

February 2015 ns-3.22 released : ns-3.22 provides a number of updates related to WiFi su...

November 2014 2015 Workshop on ns-3 announced : The NS-3 Consortium is organizing the 2015 edition of t...

How the project operates

- Project provides three annual software releases
- Users interact on mailing lists and using Bugzilla bug tracker
- Code may be proposed for merge
 - Code reviews occur on a Google site
- Maintainers (one for each module) fix or delegate bugs, participate in reviews
- Project has been conducting annual workshop and developer meeting around SIMUTools through 2013
 - Some additional meetings on ad hoc basis
- Summer projects (Google Summer of Code, ESA Summer of Code in Space, others...)

Maintainers, Authors, Users

- ~10-15 maintainers at any given time
- 191 authors credited in AUTHORS file
- Over 6000 subscribers to ns-3-users Google Groups forum
- Over 1500 subscribers to ns-developers mailing list
- Various project forks exist (on Github and elsewhere)

Contributed code and associated projects

The screenshot shows the 'Overall ndnSIM documentation' page. It features a sidebar with navigation links like 'Introduction', 'Support', and 'Getting Started'. The main content area is titled 'Welcome to ndnSIM NS-3 based NDN simulator' and includes a 'Contents' section with a detailed list of topics such as 'Introduction', 'Getting Started', 'ndnSIM helpers', and 'Content Store'.

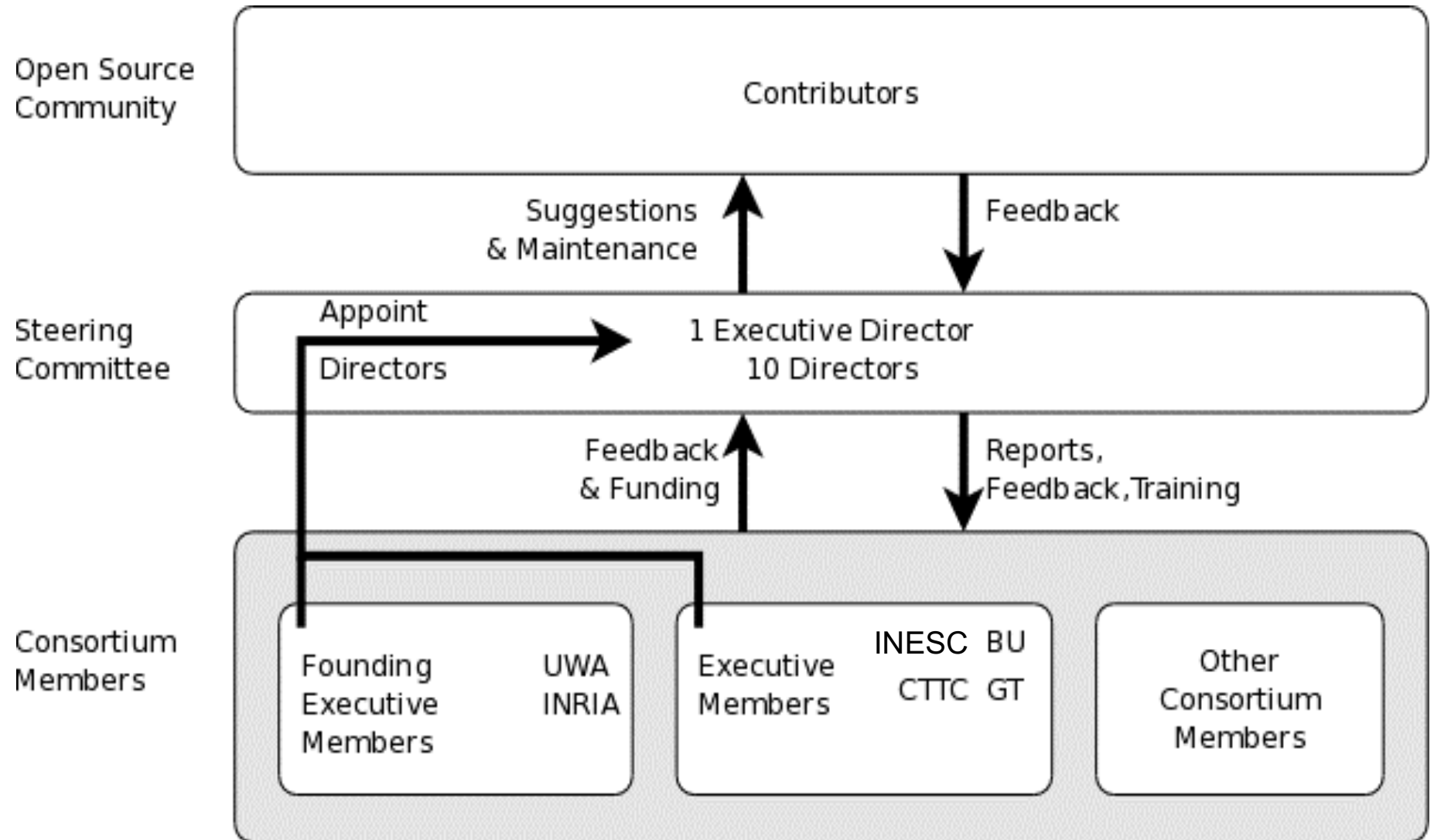
The screenshot displays the 'mptcp-ns3' project page, which implements multipath TCP on ns-3. It includes sections for 'Project description', 'Current Status', and 'Getting Started'. The 'Project description' section states that the project focuses on developing a Multipath TCP implementation for research purposes. The 'Current Status' section mentions that the implementation is close to the MPTCP specification and lists various options like MPC, ADD, REMOVE, JOIN, etc. The 'Getting Started' section provides a link to the wiki page for instructions on running simulations.

The screenshot shows the website of the Karlsruhe Institute of Technology (KIT) Decentralized Systems and Network Services Research Group. It features a sidebar with navigation links like 'Home', 'News', 'Teaching', and 'Research'. The main content area is titled 'PhySimWiFi for NS-3' and includes an 'Overview' section that describes the implementation of the OFDM-based IEEE 802.11 standard within the popular network simulator NS-3.

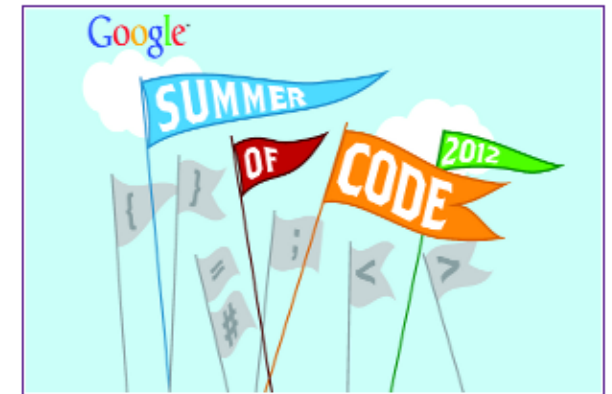
Sustainment

- The NS-3 Consortium is a collection of organizations cooperating to support and develop the ns-3 software.
- It operates in support of the open source project
 - by providing a point of contact between industrial members and ns-3 developers,
 - by sponsoring events in support of ns-3 such as users' days and workshops,
 - by guaranteeing maintenance support for ns-3's core, and
 - by supporting administrative activities necessary to conduct a large open source project.

ns-3 Consortium governance



Acknowledgment of support



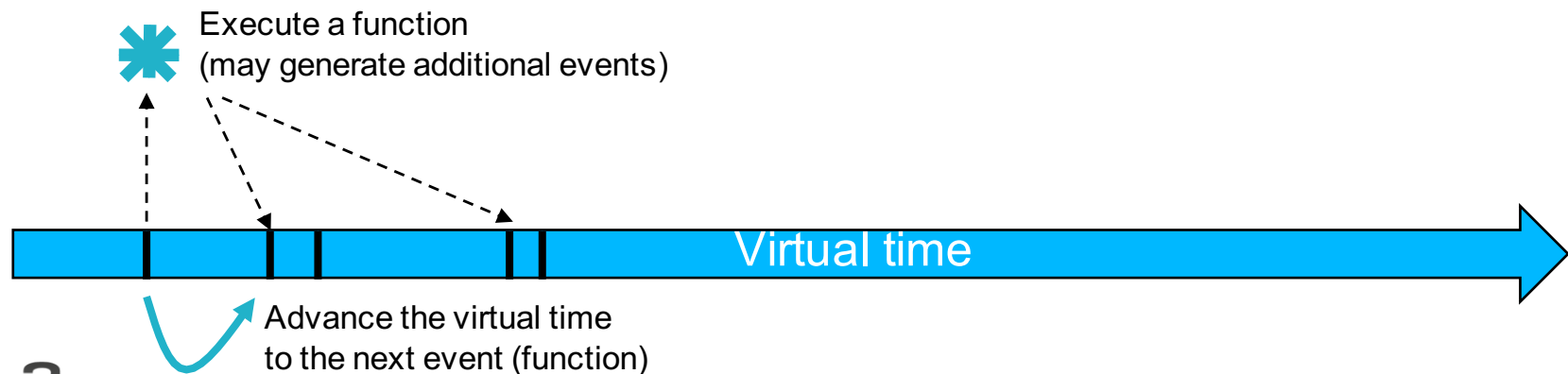
-
- Software overview

Software overview

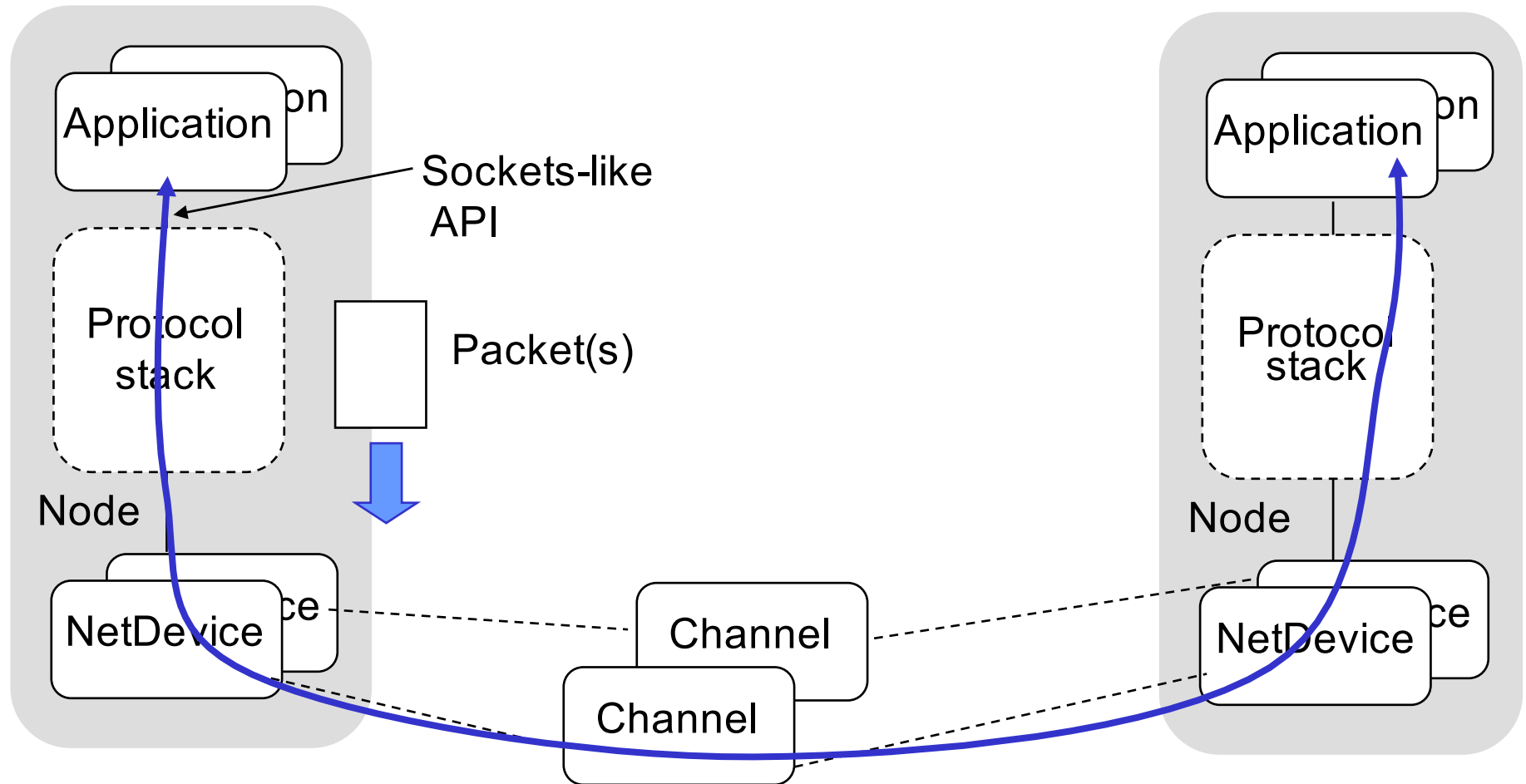
- ns-3 is written in C++, with bindings available for Python
 - simulation programs are C++ executables or Python programs
 - ~350,000 lines of C++ (cloc estimate)
 - almost exclusively C++98, beginning to use C++11
- ns-3 is a GNU GPLv2-licensed project
- ns-3 is mainly supported for Linux, OS X, and FreeBSD
 - Windows Visual Studio port available
- ns-3 is not backwards-compatible with ns-2

Discrete-event simulation basics

- Simulation time moves in discrete jumps from event to event
- C++ functions schedule events to occur at specific simulation times
- A simulation scheduler orders the event execution
- `Simulation::Run()` executes a single-threaded event list
- Simulation stops at specific time or when events end



The basic ns-3 architecture



Software orientation

Key differences from other network simulators:

- 1) Command-line, Unix orientation
 - vs. Integrated Development Environment (IDE)
- 2) Simulations and models written directly in C++ and Python
 - vs. a domain-specific simulation language

ns-3 does not have a graphical IDE

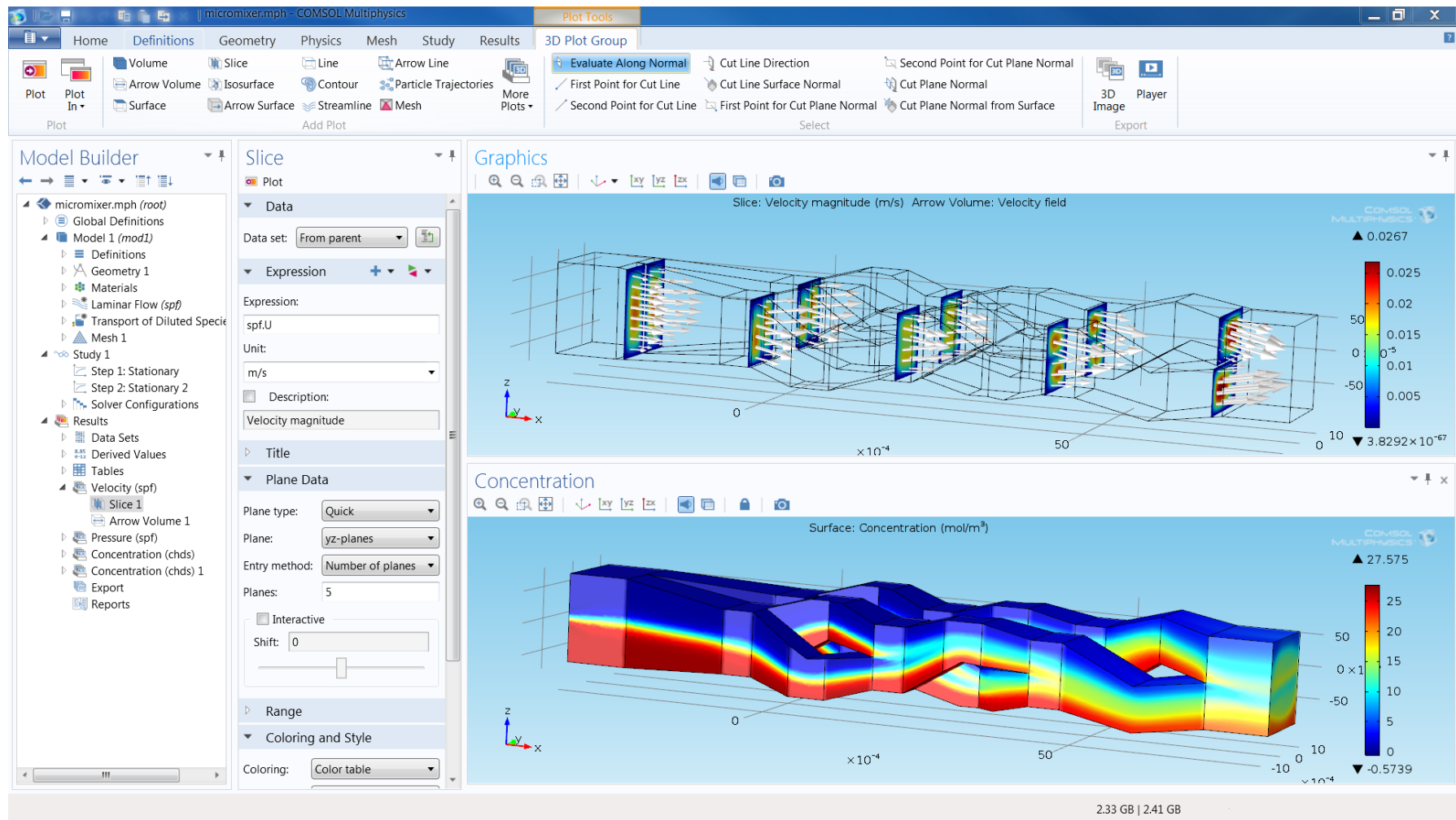


Figure source: <https://www.comsol.com/comsol-multiphysics>

ns-3 Training, June 2017

ns-3 not written in a high-level language

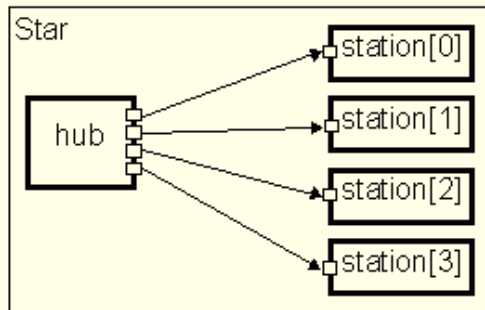
Submodule vectors, gate vectors and multiple connections are illustrated in the following example:

```
simple Hub
  gates:
    out: output[];
endsimple

simple Station //...

module Star
  submodules:
    hub: Hub
    gatesizes: output[4];
    station: Station[4];
  connections:
    for i=0..3 do
      hub.output[i] --> station[i].in;
    endfor
endmodule
```

The result of the above is depicted in Fig.4.



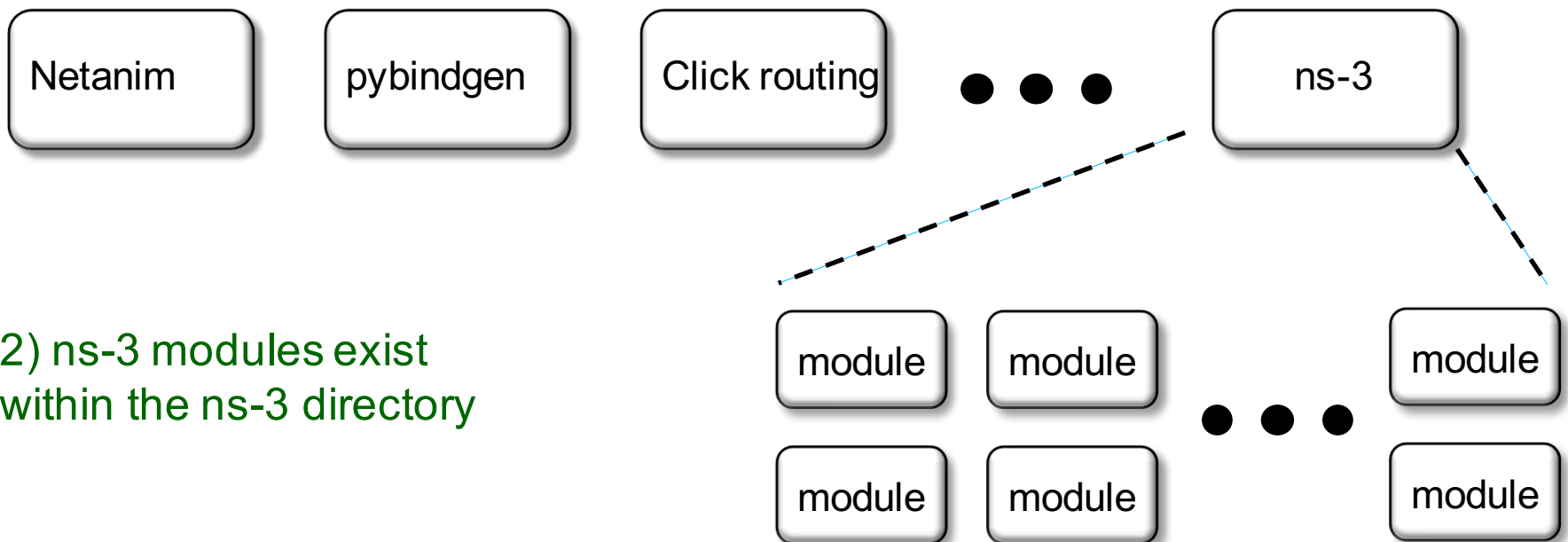
Example of OMNeT++ Network Description (NED) language

Figure excerpted from <http://www.ewh.ieee.org/soc/es/Nov1999/18/ned.htm>

Software organization

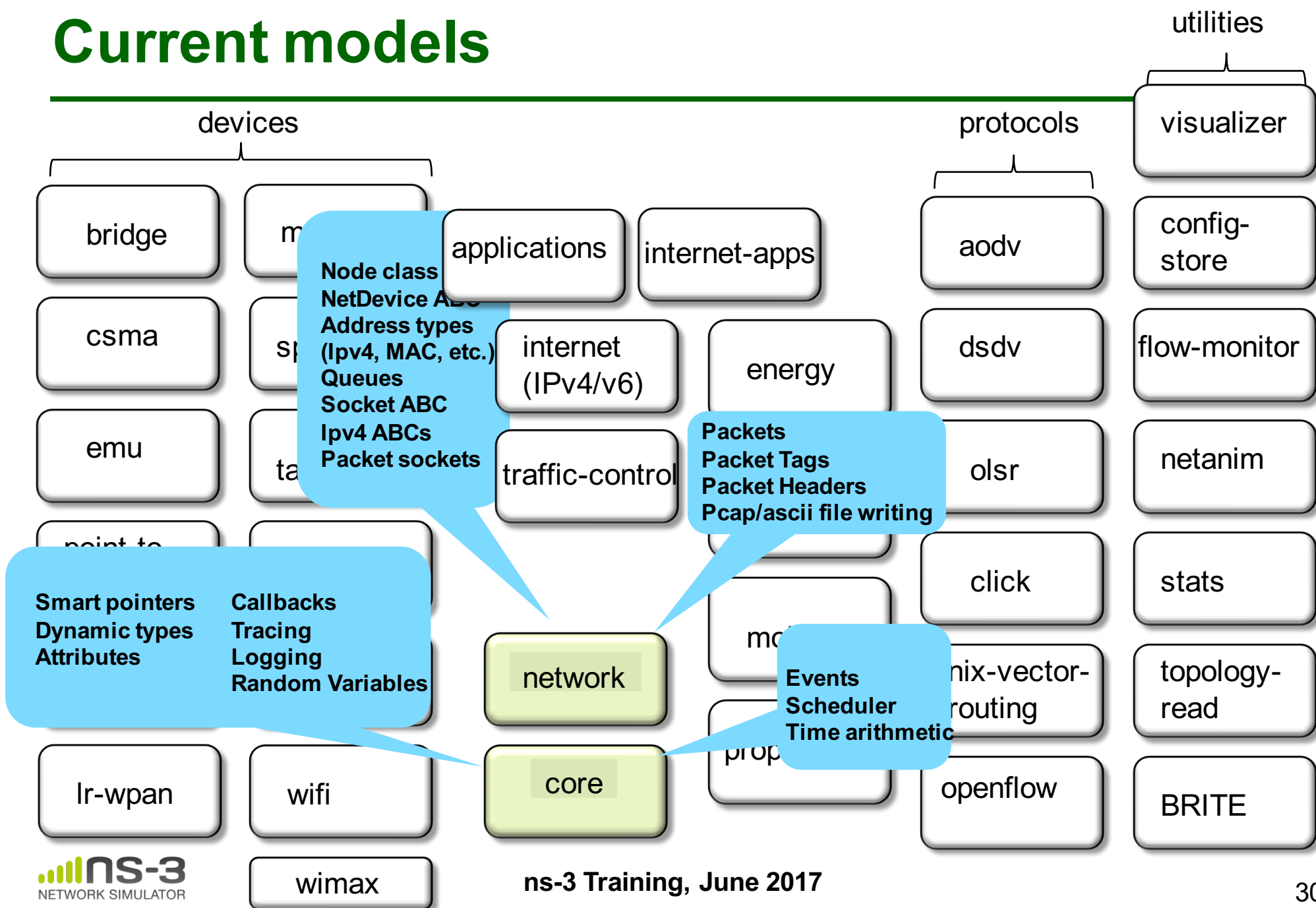
- Two levels of ns-3 software and libraries

1) Several supporting libraries, not system-installed, can be in parallel to ns-3



2) ns-3 modules exist within the ns-3 directory

Current models



Module organization

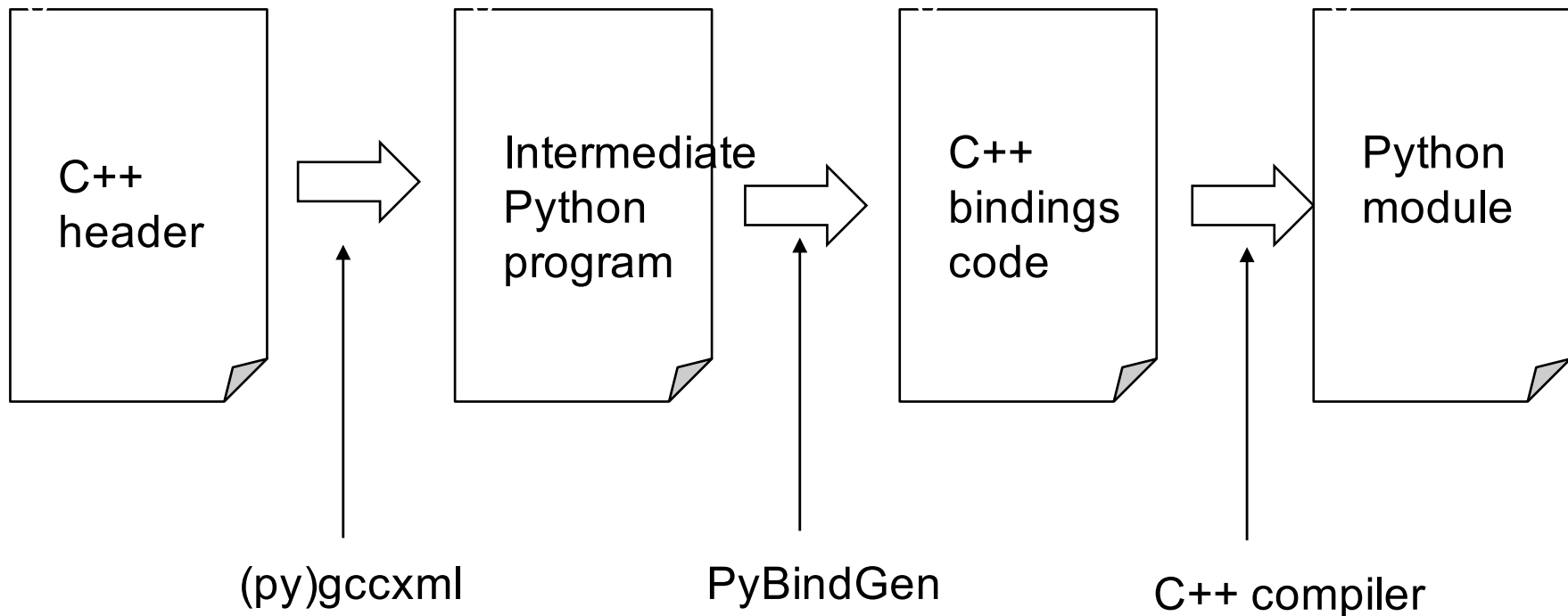
- models/
- examples/
- tests/
- bindings/
- doc/
- wscript

ns-3 programs

- ns-3 programs are C++ executables that link the needed shared libraries
 - or Python programs that import the needed modules
- The ns-3 build tool, called 'waf', can be used to run programs
- waf will place headers, object files, libraries, and executables in a 'build' directory

Python bindings

- ns-3 uses a program called PyBindGen to generate Python bindings for all libraries



Integrating other tools and libraries

Other libraries

- more sophisticated scenarios and models typically leverage other libraries
- ns-3 main distribution uses optional libraries (libxml2, gsl, mysql) but care is taken to avoid strict build dependencies
- the 'bake' tool (described later) helps to manage library dependencies
- users are free to write their own Makefiles or wscripts to do something special

Matplotlib

- `src/core/examples/sample-rng-plot.py`

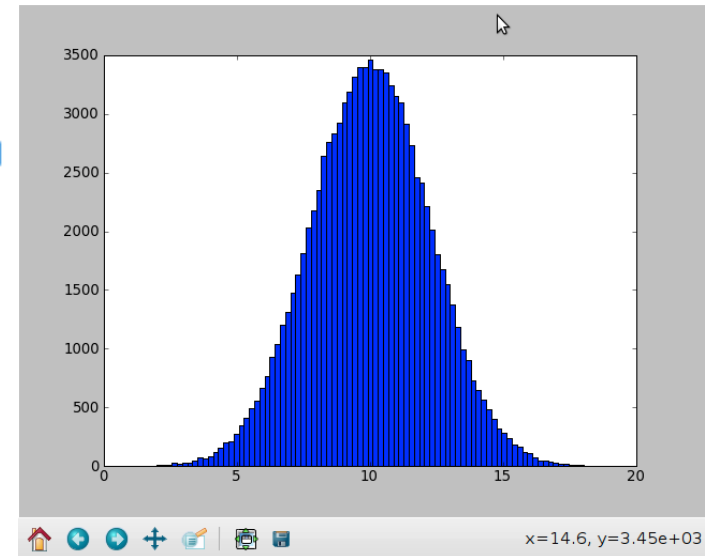
```
# Demonstrate use of ns-3 as a random number generator integrated  
# plotting tools; adapted from Gustavo Carneiro's ns-3 tutorial
```

```
import numpy as np  
import matplotlib.pyplot as plt  
import ns.core
```

```
# mu, var = 100, 225  
rng = ns.core.NormalVariable(100.0, 225.0)  
x = [rng.GetValue() for t in range(10000)]
```

```
# the histogram of the data  
n, bins, patches = plt.hist(x, 50, normed=1, facecolor='g', alpha=0.75)
```


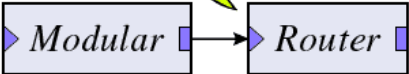
```
plt.title('ns-3 histogram')  
plt.text(60, .025, r'$\mu=100,\ \sigma=15$')  
plt.axis([40, 160, 0, 0.03])  
plt.grid(True)  
plt.show()
```



Click Modular Router

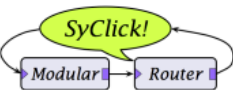

[Click!](#) [Login](#)

[Show pagesource](#) [Old revisions](#) [Sitemap](#) [Recent changes](#) [Search](#)

```
graph LR; Modular[Modular] --> Router[Router]
```

The Click Modular Router Project

```
graph LR; Modular[Modular] --> Router[Router]; Router --> SyClick!(SyClick!); SyClick! --> Modular
```

NEWS (September 24, 2011): **Click 2.0.1 released!**

SyClick: Symposium on Click Modular Router^{*} was **November 23-24, 2009, Ghent, Belgium!** An excellent time was had. Video of the presentations is now available.

This is the DokuWiki^{*} for the Click modular router. Click was originally developed at MIT^{*} with subsequent development at Mazu Networks^{*}, ICIR^{*}, UCLA^{*}, and Meraki^{*}.

OpenFlow Switch

Please Note: This website has been archived and is no longer maintained.
See the [Open Networking Foundation](#) for current OpenFlow-related information.



Google™ Custom Search

Search

Home

Videos

Documents

News

Research

About

Page Discussion View source History

MPLS with OpenFlow/SDN

Contents [hide]

- 1 Motivation
- 2 Changes to the OpenFlow protocol
- 3 Demos
- 4 Publications
- 5 Talks
- 6 People

Motivation

MPLS networks have evolved over the last 10-15 years to become critically important for ISPs. They provide two key services: traffic engineering in IP networks and L2 or L3 enterprise VPNs. However as carriers deploy MPLS networks, they find that (a) even though the MPLS data plane was meant to be simple, vendors end up supporting MPLS as an additional feature on complex, energy hogging, expensive core routers; and (b) the IP/MPLS control plane has become exceedingly complex with a wide variety of protocols tightly intertwined with the associated data-plane mechanisms.

Quick Navigation

- » [OpenFlow Specs](#)
- » [Bug Tracking](#)
- » [Wiki](#)
- » [Legal](#)
- » [Log in](#)

OpenFlow White Paper



Download the OpenFlow Whitepaper (PDF)

OpenFlow Specification



Download v1.1.0 Implemented (PDF)

Wiki Tools


Personal tools

- » [Log in](#)

Navigation

- » [OpenFlow.org](#)

CORE emulator




Networks and Communication Systems Branch

- Focus Areas
- Projects
- Products
- Organization

/ NRL / ITD / NCS / Common Open Research Emulator (CORE)

Common Open Research Emulator (CORE)

The Common Open Research Emulator (CORE) is a tool for emulating networks on one or more machines. You can connect these emulated networks to live networks. CORE consists of a GUI for drawing topologies of lightweight virtual machines, and Python modules for scripting network emulation.




NCS Home

Focus Areas

Projects

Products

Organization



mininet emulator

The screenshot shows the GitHub repository for mininet. The repository is public and has 468 stars and 204 forks. The page title is "Link modeling using ns 3". The contents list includes:

- Introduction
 - ns-3 emulation features
 - Link simulation with ns-3
- Details
 - How to achieve communication of ns-3 process with TAP interfaces in distinct namespaces?
 - Architecture: single ns-3 thread or multiple processes?
- Code
 - Mininet
 - ns-3 patches

The right sidebar contains a table of contents with links to Mininet, Get Started, Sample Workflow, Walkthrough, Overview, Download, Documentation, Videos, Source Code, Apps, FAQ, Wiki, Teaching, Papers, and GSoC 2013.

Co-simulation frameworks have emerged

- PNNL's FNCS framework integrates ns-3 with transmission and distribution simulators

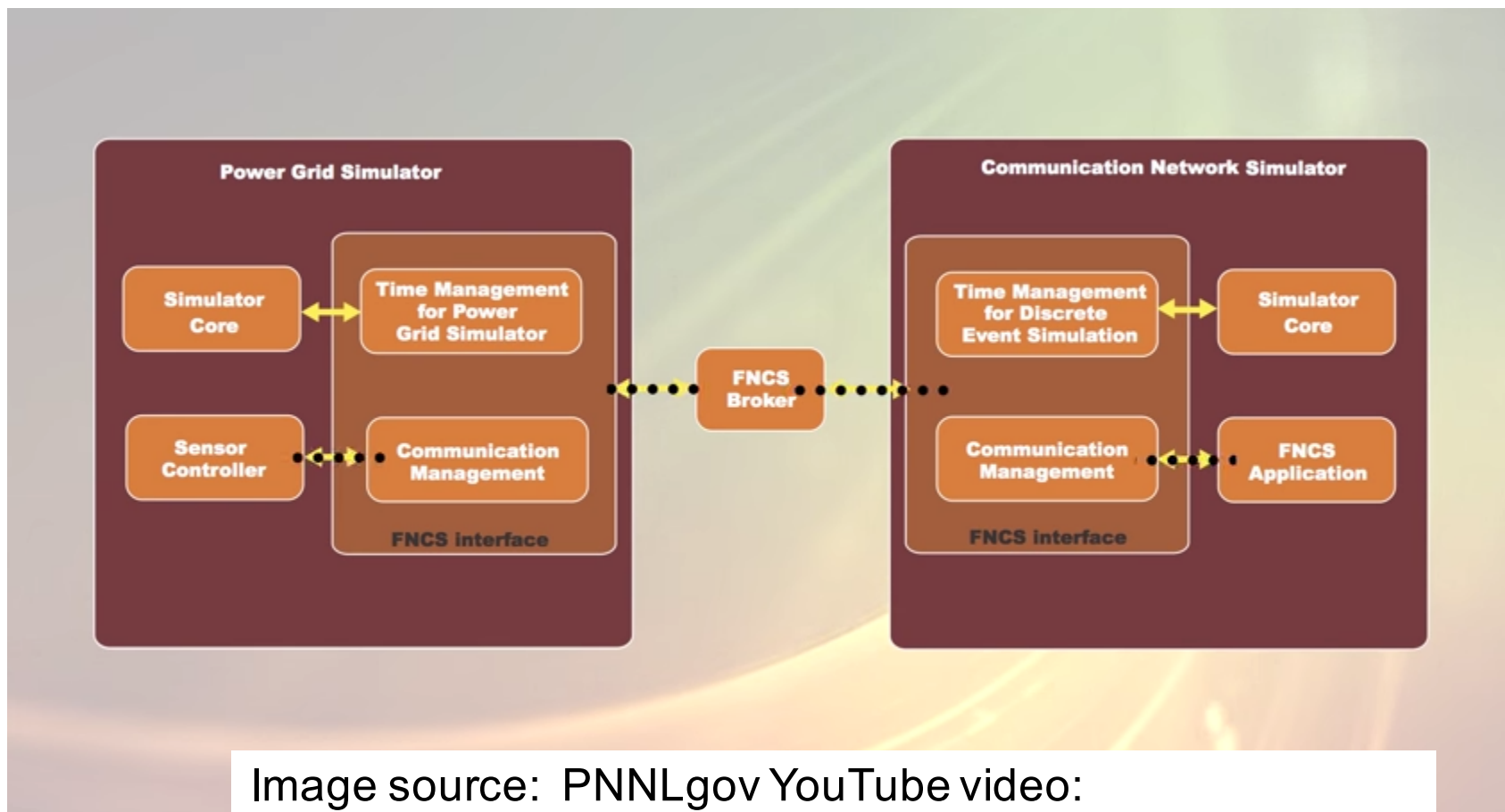


Image source: PNNLgov YouTube video:
Introducing FNCS: Framework for Network Co-Simulation
ns-3 Training, June 2017

FAQs

- Does ns-3 have a Windows version?
 - Yes, for Visual Studio 2012
 - http://www.nsnam.org/wiki/Ns-3_on_Visual_Studio_2012
- Does ns-3 support Eclipse or other IDEs?
 - Instructions have been contributed by users
 - http://www.nsnam.org/wiki/HOWTO_configure_Eclipse_with_ns-3
- Is ns-3 provided in Linux or OS X package systems (e.g. Debian packages)?
 - Not officially, but some package maintainers exist
- Does ns-3 support NRL protolib applications?
 - Not yet

Summarizing

- ns-3 models are written in C++ and compiled into libraries
 - Python bindings are optionally created
- ns-3 programs are C++ executables or Python programs that call the ns-3 public API and can call other libraries
- ns-3 is oriented towards the command-line
- ns-3 uses no domain specific language
- ns-3 is not compatible with ns-2